VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra telekomunikační techniky

Automatizované generování modelů v systému Scilab/Scicos dle výsledků výpočtů v prostředí Scilab

Automated Scilab/Scicos Model Generation Based on Scilab Design

2017 Daniel Hyka

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra telekomunikační techniky

Zadání bakalářské práce

Student: **Daniel Hyka**

Studijní program: B2647 Informační a komunikační technologie

Studijní obor: 2601R013 Telekomunikační technika

Téma: Automatizované generování modelů v systému Scilab/Scicos dle výsledků výpočtů v prostředí Scilab
Automated Scilab/Scicos Model Generation Based on Scilab Design

Jazyk vypracování: čeština

Zásady pro vypracování:

Cílem práce je popsat strukturu "xcos" souboru systému Scilab/Scicos a dále popsat možnosti jeho automatického generování pro známé struktury modelů dle výsledků výpočtů v systému Scilab.

1. Popište rozložení pracovní plochy systému Scilab včetně významu jednotlivých prvků. Popište, jak spustit prostředí Scicos a popište jednotlivé prvky pracovního prostředí.
2. Na jednoduchých modelech popište dílčí části "xcos" souboru.
3. Popište detailně obecnou strukturu "xcos" souboru.
4. Vytvořte funkci pro generování "xcos" souborů pro simulaci digitálních filtrů dle výsledků návrhu v systému Scilab.
5. Srovnejte výsledky simulací automaticky generovaných a ručně vytvořených modelů.

Práce bude vytvořena v typografickém systému LaTeX.

Seznam doporučené odborné literatury:

[1]UHLÍŘ, Jan a Pavel SOVKA. *Číslicové zpracování signálů*. Vyd. 2. přeprac. Praha: Vydavatelství ČVUT, 2002, 327 s. ISBN 80-01-02613-2.

[2]Stephen L. Campbell, Jean-Philippe Chancelier, Ramine Nikoukhah. *Modeling and Simulation in Scilab/Scicos*. 2006. ISBN-10: 0-387-27802-8 ISBN-13: 978-0387278025.

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **Ing. Jan Skapa, Ph.D.**

Datum zadání: 01.09.2015

Datum odevzdání: 28.04.2017

doc. Ing. Miroslav Vozňák, Ph.D.
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 28.Dubna 2017 ..... Dyka

Tímto bych chtěl poděkovat svému vedoucímu bakalářské práce, Ing. Janu Skapovi, Ph.D., za rady, trpělivost, připomínky a náměty.

Abstrakt

Cílem bakalařské práce je vytvořit program, který bude převádět bloky a spoje z programu Simulink do programu Xcos. Xcos je nástavbou programu Scilab, který je volně dostupným programem pro numerické výpočty a alternativou licencovaného programu MATLAB. Práce bude zahrnovat popis pracovní plochy programu Scilab a Xcos a také detailní popis souboru Xcos, který je potřeba k pochopeni jak funguje převod. Součástí práce bude také porovnání zkušebního zapojení v simulinku a převedeného zapojení v Xcos.

Klíčová slova: Xcos, Scilab, MATLAB, Simulink

Abstract

The object of my work is creating of program which converts blocks and links from Simulink into Xcos. Xcos is extension of program Scilab which is open source tool for numerical calculations and great alternative to licensed program MATLAB. The work will include description of graphical user interface of Scilab and Xcos programs and also detailed description of Xcos files which are needed for understanding how conversion works. The work will contain comparison of example in Simulink with converted version in Xcos.

Keywords: Xcos, Scilab, MATLAB, Simulink

Seznam použitých zkratek a symbolů

CSV	–	Comma-Separated Values
MATLAB	–	MATrix LABoratory
URL	–	Uniform Resource Locator
XML	–	eXtensible Markup Language

Obsah

Úvod **3**

1 Popis rozložení pracovní plochy Scilab a Xcos **4**
1.1 Rozložení pracovní plochy Scilab 4
1.2 Popis okna prohlížeče souborů 4
1.3 Popis okna prohlížeče proměnných 6
1.4 Popis okna historie příkazů 8
1.5 Popis okna konzole 9
1.6 Popis editoru SciNotes 10
1.7 Rozložení pracovní plochy Xcos 12
1.8 Popis palety Xcosu 12
1.9 Popis pracovní plochy Xcos 13

2 Detailní popis dílčích části "xcos"souboru na jednoduchých modelech **15**
2.1 Detailní rozbor souboru .xcos obsahující jeden blok 15
2.2 Detailní rozbor souboru .xcos obsahující dva bloky 17
2.3 Detailní rozbor souboru .xcos obsahující tři bloky a uzel 20

3 Tvorba modelů pomocí jazyka Scilab **23**
3.1 Popis tvorby jednoduchého zapojení s uzlem 23
3.2 Tvorba bloku CLOCK_f 27

4 Automatické generování bloků z programu Simulink do programu Scilab **30**
4.1 Rozbalení a načtení souboru Simulink (.slx) 30
4.2 Generování bloku GAINBLK_f 31
4.3 Generování bloku BIGSOM_f 32
4.4 Generování bloku GENSIN_f 33
4.5 Generování bloku CMSCOPE 34
4.6 Generování spojů 35
4.7 Porovnání zkušebního zapojení s převedeným zapojením 37

Závěr **38**

Literatura **39**

Seznam obrázků

1 Pracovní plocha Scilab . . . 4
2 Prohlížeč souborů . . . 5
3 Tlačítka pro prohlížeč souborů . . . 5
4 Menu pro soubory . . . 6
5 Prohlížeč proměnných . . . 7
6 Tlačítka pro prohlížeč proměnných . . . 7
7 Editor proměnných . . . 7
8 Zobrazení všech tlačítek pro editor proměnných . . . 8
9 Historie příkazů . . . 8
10 Tlačítka historie příkazů . . . 9
11 Konzole . . . 9
12 Menu konzole . . . 9
13 Tlačítka konzole . . . 10
14 SciNotes při spuštění . . . 10
15 Menu SciNotes . . . 11
16 Tlačítka SciNotes . . . 11
17 Pracovní plocha Xcos . . . 12
18 Xcos paleta . . . 12
19 Menu složky palette . . . 13
20 Xcos editor . . . 13
21 Menu Xcos . . . 14
22 Tlačítka Xcos . . . 14
23 Blok GENSIN_f . . . 15
24 Tabulka dat bloku GENSIN_f . . . 16
25 Blok GENSIN_f a CSCOPE se spojem . . . 17
26 Tabulka dat bloku CSCOPE . . . 18
27 Zapojení tří bloků uzlem . . . 20
28 Vyobrazení bodu na spoji . . . 22
29 Blokové zapojení . . . 23
30 Blokové zapojení pro blok CLOCK_f . . . 27
31 Soubory po rozbalení souboru .slx . . . 30
32 Struktura Gain . . . 31
33 Struktura Sum . . . 32
34 Struktura Sine wave . . . 33
35 Struktura TimeScope . . . 34
36 Struktura Line . . . 35
37 Struktura SID . . . 36
38 Porovnání: vlevo Simulink vpravo Xcos . . . 37

Úvod

Tato práce se zabývá možností převodu bloků z licencovaného programu Simulink, který je nástavbou programu MATLAB do volně dostupného programu Xcos, který je nástavbou programu Scilab. Dále se práce zaměřuje na detailní rozbor pracovních prostředí Scilab, Xcos a práci s Xcos soubory.

V první kapitole je detailně popsané pracovní prostředí programu Scilab, Xcos. Jsou zde popsány všechny možnosti oken v pracovním prostředí, dále pak všechna menu, klávesové zkratky, rychlá funkční tlačítka.

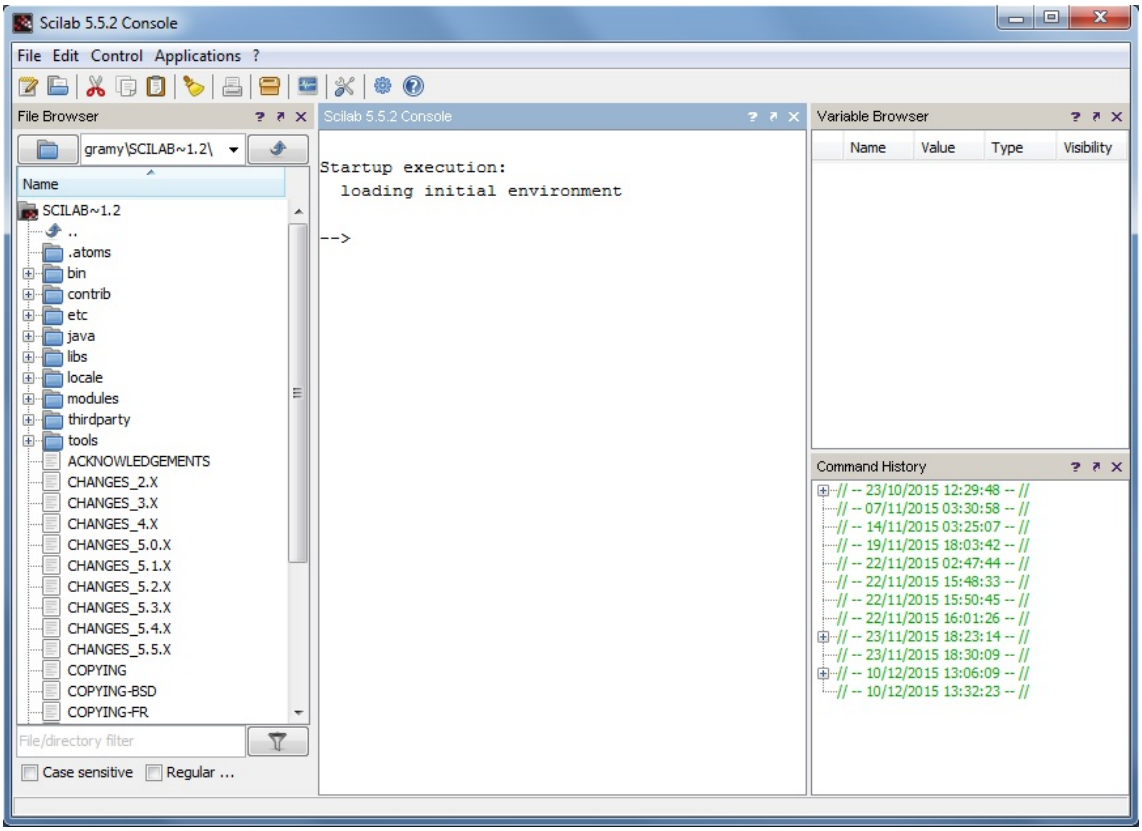
V druhé kapitole je detailní rozbor Xcos souboru na jednoduchých modelech. Podkapitoly jsou rozděleny podle počtu bloků a u každého bloku je pak zaměření na stěžejní část kódu. Jsou zde popsány rozdíly mezi jedním blokem a modelem obsahující vice bloků.

Třetí kapitola je zaměřena na tvorbu bloků pomoci jazyka Scilab. Je zde vysvětleno jak se tvoří Xcos diagram, grafická část bloku a nakonec modelová část bloku. Dále je zde vysvětleno jak se tvoří spoje mezi jednotlivými bloky a uzel pro spoj mezi více bloky. Jsou zde ukázány rozdíly ve tvorbě bloku sampleCLK a klasickém bloku CLOCK_f.

Čtvrtá kapitola obsahuje podrobný popis tvorby programu. Je zde ukázáno jak se Simulink soubor načítá do vlastní struktury. Dále jsou zde ukázány stěžejní části kódu pro jednotlivé bloky a spoje. V poslední řadě je zde porovnání mezi výstupem programu a zadaným Simulink zapojením.

1 Popis rozložení pracovní plochy Scilab a Xcos

1.1 Rozložení pracovní plochy Scilab

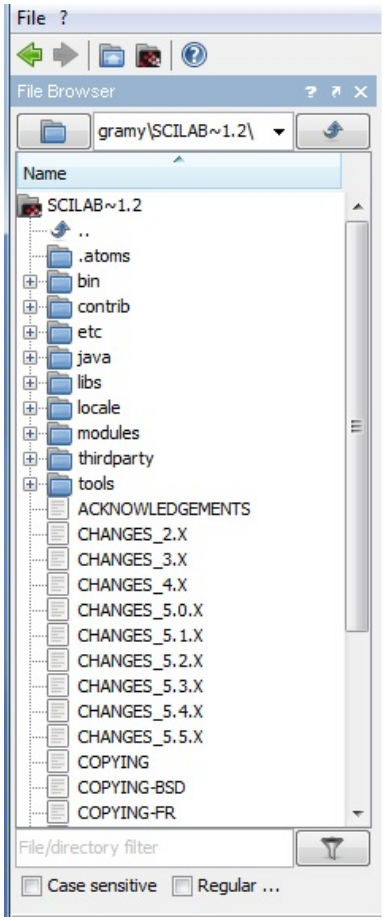


Obrázek 1: Pracovní plocha Scilab

Pracovní plocha programu Scilab se rozděluje do čtyř částí, kterými jsou konzole, prohlížeč souboru, prohlížeč proměnných a historie příkazů. Každá z těchto částí má po aktivaci okna své vlastní ikony a menu. Prostředí si můžeme dle potřeby upravovat, můžeme okna zmenšovat či zvětšovat, nepotřebná okna můžeme vypnout a nebo si okno otevřít přes celou obrazovku jako samostatný oddíl.

1.2 Popis okna prohlížeče souborů

V této části pracovní plochy Scilab se orientujeme v adresářích. Můžeme zde rychle spustit soubor ve Scilab nebo načíst soubor do programu Xcos, či uložit soubor do zobrazeného adresáře. Dále zde můžeme soubory editovat v libovolném textovém prohlížeči.



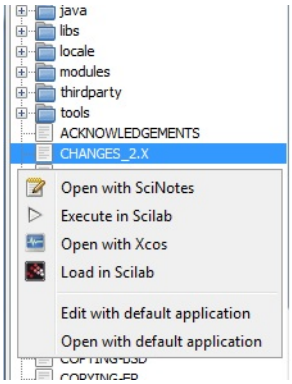
Obrázek 2: Prohlížeč souborů

Na obrázku 2 můžeme vidět menu file, ve kterém je pouze tlačítko pro zavření okna prohlížeče souborů. Vedle menu file je otazník, který nám otevře celkovou nápovědu programu Scilab.

Obrázek 3: Tlačítka pro prohlížeč souborů

Na obrázku 3 jsou vyznačena funkční tlačítka pro prohlížeč souborů. Zelené šipky slouží pro orientaci v adresářích zpět a vpřed, další tlačítko slouží pro návrat do domovského adresáře. Čtvrté tlačítko nás okamžitě přesune do adresáře s nainstalovaným programem Scilab. Tlačítko otazníku nám otevře nápovědu programu Scilab.
Bílá tlačítka v modrém poli nám slouží pro úpravu celého okna. Otazník nám otevře nápovědu pro tuto určitou část a nemusíme tak hledat v celé nápovědě. Ikona šipky nám

otevře okno jako další program a oddělí se tak z klasické pracovní plochy Scilabu. Pro vložení zpět do pracovní plochy Scilab musíme kurzorem najet na tuto modrou část a přetáhnout na pracovní plochu. Tlačítko křížku nám prohlížeč souborů vypne. Pokud prohlížeč souborů vypneme pomoci zmiňovaného křížku, opět ho zapneme v konzoli v menu applications, ale o tom si řekneme v další kapitole.
Modrá složka nám otevře okno pro rychlý výběr adresáře, ve kterém chceme spravovat soubory. Vedle složky vidíme aktuální adresář ve kterém se nacházíme. Šipka nahoru nás vrátí o adresář zpět.



Obrázek 4: Menu pro soubory

Na obrázku 4 můžeme vidět kontextové menu pro vybraný soubor. Můžeme ho otevřít hned pomoci několika programů. Pomocí SciNotes se nám otevře pro programovou úpravu, ale o SciNotes si ještě povíme v další části, a nebo soubor můžeme ihned spustit pomocí Scilab nebo Xcos. Soubor můžeme také spustit defaultním programem který k němu máme přiřazený, nebo jej upravovat.

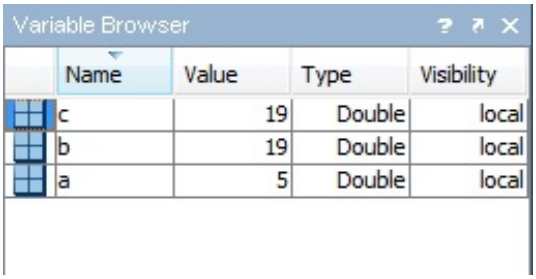
1.3 Popis okna prohlížeče proměnných

V této části pracovní plochy Scilab můžeme vidět v tabulce všechny zadané a vypočtené hodnoty, které v programu zadáme. Dále můžeme vidět v jakém typu jsou hodnoty uložené. Z uložených hodnot můžeme vytvořit rychlý graf, nebo můžeme hodnoty exportovat do souboru CSV.

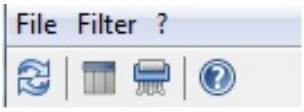
Na obrázku 5 vidíme prohlížeč proměnných, kde jsou hodnoty zapsané do tabulky. proměnné jsou zapisované po řádcích. Sloupce je rozdělují do čtyř kategorii podle jména, hodnoty kterou obsahují, typu(například string, double) a viditelnosti.

Bílá tlačítka v modrém poli mají na všech částech stejný význam jako u prohlížeče souborů. V dalších kapitolách už se o nich zmiňovat nebudeme.

Na obrázku 6 můžeme vidět všechna tlačítka a menu pro okno proměnných. V menu file nalezneme pouze tlačítka na aktualizaci proměnných a ukončení okna proměnných. V menu filter si můžeme odfiltrovat pouze proměnné, které chceme sledovat a které nechceme zobrazovat.

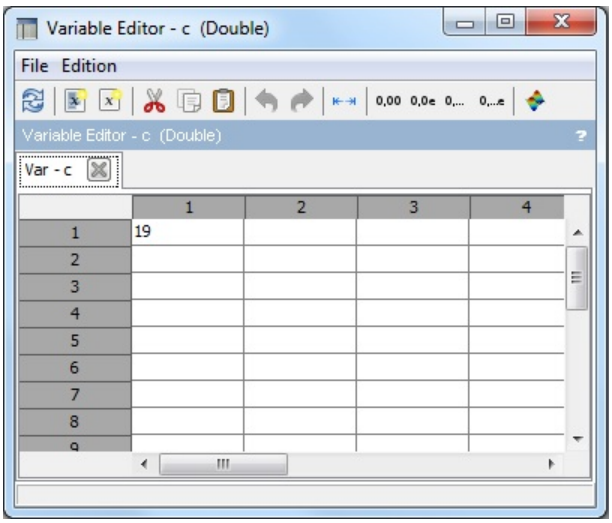


Obrázek 5: Prohlížeč proměnných



Obrázek 6: Tlačítka pro prohlížeč proměnných

První tlačítko nám slouží pro aktualizaci proměnných stejně jako v menu file. Druhé tlačítko slouží pro modifikaci proměnných, ke které si něco řekneme v dalším odstavci. Třetím tlačítkem můžeme proměnou z tabulky smazat, posledním tlačítkem je nápověda.



Obrázek 7: Editor proměnných

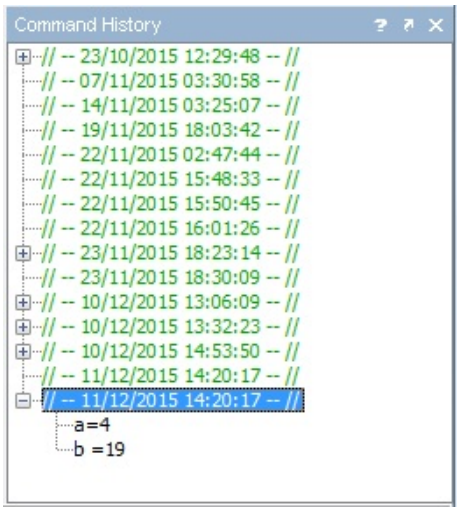
Na obrázku 7 můžeme vidět editor proměnných, který jsme spustili pomoci modify u prohlížeče proměnných také lze spustit dvojitým poklepáním na proměnou. Toto okno nám slouží pro úpravu a vytváření proměnných. Máme možnost ukládat do souboru CSV a vytvářet grafy různých typů.

Obrázek 8: Zobrazení všech tlačítek pro editor proměnných

1- Tlačítko aktualizace proměnných
2- Tlačítko pro vytvoření nové proměnné z označených hodnot
3- Tlačítko vytvoření nové proměnné
4- Tlačítka pro vyjmutí/kopírování/vložení proměnných
5- Tlačítko zpět
6- Tlačítko vpřed
7- Tlačítko roztažení sloupce při větších hodnotách
8- Tlačítka pro formát zapsání čísla
9- Tlačítko pro vytvoření vybraného grafu

1.4 Popis okna historie příkazů

V této části pracovní plochy Scilab můžeme vidět všechny zadané příkazy co jsme použili a v jaký den jsme je použili. Příkazy můžeme tak rychle znovupoužít. Historie příkazů se nám ukládá do souboru, který se zvolíme v nastavení a můžeme tak historii jednoduše načíst ze souboru.



Obrázek 9: Historie příkazů

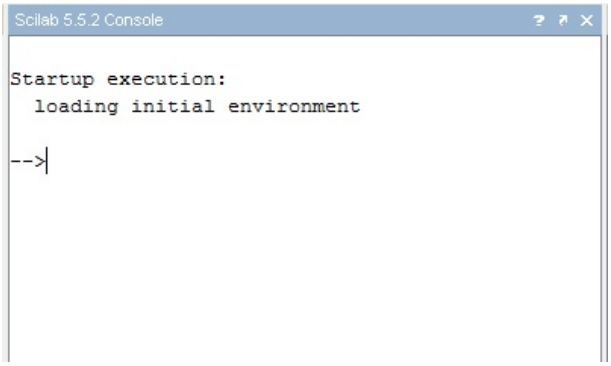
Na obrázku 9 vidíme okno s historii příkazů, kde je zelenou barvou zapsáno datum a čas spuštění programu. Po zmáčknutí znaménka plus vedle data se nám rozbalí všechny příkazy co jsme použili za toto spuštění a dvojklikem je jednoduše znovupoužít.

Obrázek 10: Tlačítka historie příkazů

Na obrázku 10 vidíme všechna tlačítka a menu pro historii příkazů. V menu file nalezneme pouze tlačítko pro vypnuti okna historie, nebo také můžeme použít klávesovou zkratku ctrl+W pro vypnutí. V menu edit máme možnost vyjmutí nebo kopírovaní příkazů, dále zde nalezneme možnost smazaní určitého dne a nebo celé historie. Tlačítka rychlého použití, která jsou pod menu mají stejnou funkci jako v menu edit, na konci tlačítek rychlého použití nalezneme opět nápovědu.

1.5 Popis okna konzole

Tato část pracovní plochy Scilab je nejdůležitější. Zde zadáváme naše proměnné a příkazy, které se nám zapisují do tabulky proměnných a historie. Výsledky výpočtů se ukazují přímo v konzoli. Při přidání středníku na konec příkazu se nezobrazí výsledek ale zapíše se do tabulky proměnných. Konzole je stejná jako v ostatních skriptovacích jazycích, například MATLAB. Náhled na konzoly můžeme vidět na obrázku 11.

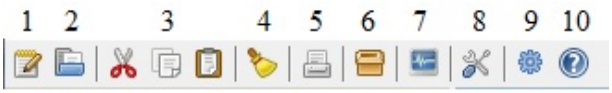


Obrázek 11: Konzole

Obrázek 12: Menu konzole

Na obrázku 12 vidíme všechna menu pro konzoly. V menu file nalezneme potřebné funkce pro práci se soubory jako jsou načtení souboru, uložení souboru, můžeme zde měnit adresář v prohlížeči souborů, dále si zde můžeme upravovat velikost stránky a nastavit možnosti tisku. Další z menu je edit ve, kterém jsou funkce jako kopírovat, vyjmout, vložit, vyčistit konzoly a historii. Menu control nám slouží pro práci s bežícím programem, který můžeme spustit, vypnout nebo přerušit. V poslední řadě tu máme

menu applications ve kterém můžeme spustit nástavby pro Scilab jako jsou SciNotes, Xcos a nebo přeložit kódu z MATLAB na Scilab kód. Dále tu máme možnost zapnutí oken prohlížeče souboru, prohlížeče proměnných a historii příkazů, pokud jsme je předtím zavřeli.

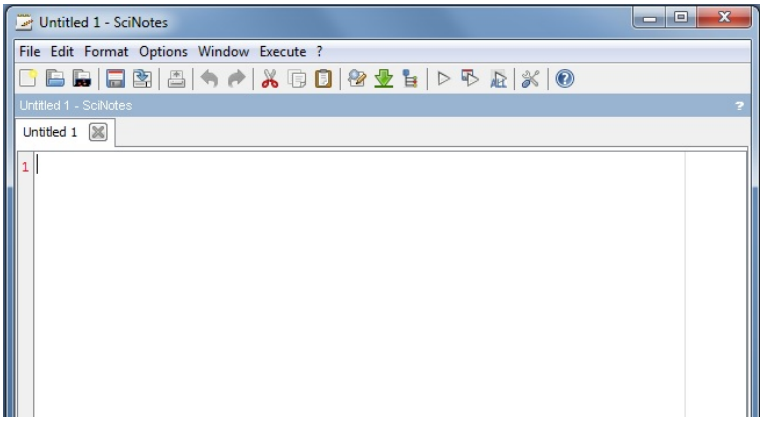


Obrázek 13: Tlačítka konzole

1- Tlačítko spuštění SciNotes
2- Tlačítko pro otevření souboru
3- Tlačítka pro vyjmutí/kopírování/vložení
4- Tlačítko vyčištění konzole
5- Tlačítko pro tisk
6- Tlačítko modul manager - ATOMS
7- Tlačítko spuštění Xcosu
8- Tlačítko spuštění nastavení
9- Tlačítko spuštění připravených demonstrací
10- Tlačítko nápovědy

1.6 Popis editoru SciNotes

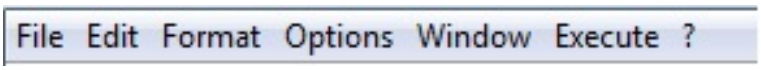
V minulé kapitole jsme si řekli jak se SciNotes spouští, v této kapitole si řekneme něco o tom jak se používá a k čemu slouží.



Obrázek 14: SciNotes při spuštění

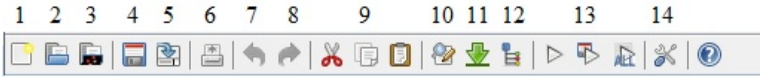
Na obrázku 14 vidíme základní obrazovku SciNotes. SciNotes je textovým editorem programu Scilab. V textovém editoru SciNotes je možno psát příkazy pod sebou stejně jak v programovacích jazycích jako jsou například Java nebo C/C++. Poté můžeme celý

program uložit a zkompilovat, nemusíme tak celý kód psát po příkazech do konzole.



Obrázek 15: Menu SciNotes

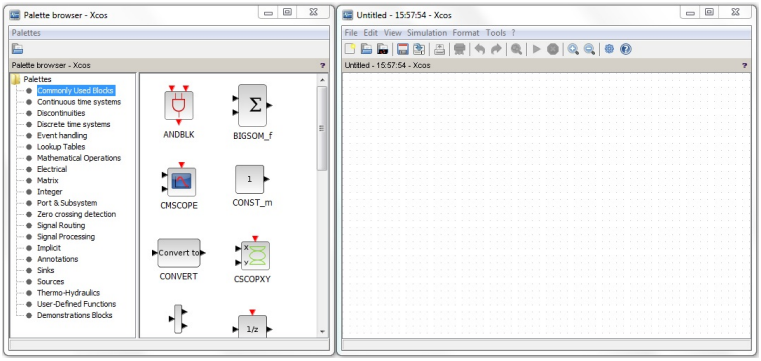
Na obrázku 15 můžeme vidět všechna dostupná menu pro SciNotes. V menu file najdeme možnosti pro práci se souborem jako například načtení souboru, uložení souboru, otevření souboru přes URL, možnost tisku a ukončení programu. Menu edit nám slouží pro práci s textem stejně jako u konzole, tedy kopírování textu, vkládaní textu, mazání textu, hledání v textu. V menu format můžeme nalézt tlačítka pro úpravu textu jako jsou například vynechání prostoru před textem, správnost odsazení, odstranění koncových mezer a zakomentování či odkomentování označeného textu. V menu options si můžeme měnit kódování souboru, nastavovat velikost písma, typ písma a barvu písma, můžeme si nastavit automatické dopisování závorek a nastavit si číslování řádků. V menu window si můžeme rozdělit jedno okno na dvě menší pod sebou nebo vedle sebe, nebo můžeme zkopírovat text do nového okna. Pomocí menu execute můžeme náš program uložit a spustit.



Obrázek 16: Tlačítka SciNotes

1- Tlačítko spuštění nového okna SciNotes
2- Tlačítko pro otevření souboru
3- Tlačítka pro otevření souboru ze složky Scilab
4- Tlačítko uložit
5- Tlačítko uložit jako
6- Tlačítko tisk
7- Tlačítko zpět
8- Tlačítko vpřed
9- Tlačítka pro vyjmutí/kopírování/vložení
10- Tlačítko hledat/přepsat
11- Tlačítko postupné hledání
12- Tlačítko navigace v kódu
13- Tlačítka spuštění a uložení
14- Tlačítko nastavení

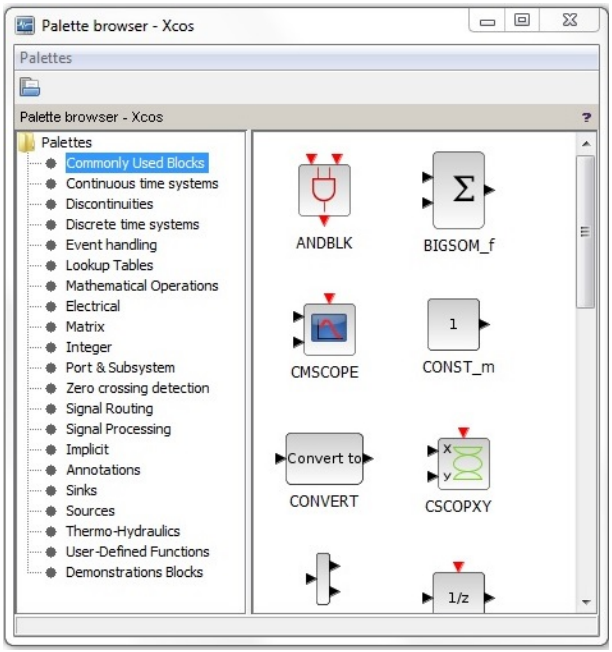
1.7 Rozložení pracovní plochy Xcos



Obrázek 17: Pracovní plocha Xcos

Pracovní plocha programu Xcos se při otevření rozkládá do dvou hlavních oken, v jednom okně máme paletu modulů, ve druhém okně máme editor do kterého bloky vkládáme a spojujeme. Na obrázku 17 můžeme vidět pracovní plochu Xcos po otevření.

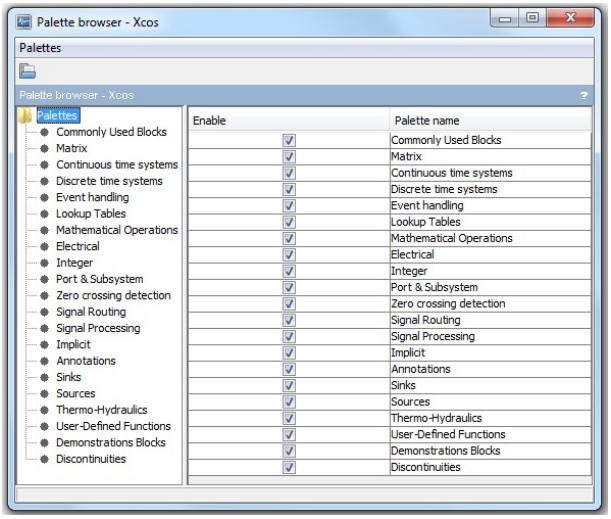
1.8 Popis palety Xcosu



Obrázek 18: Xcos paleta

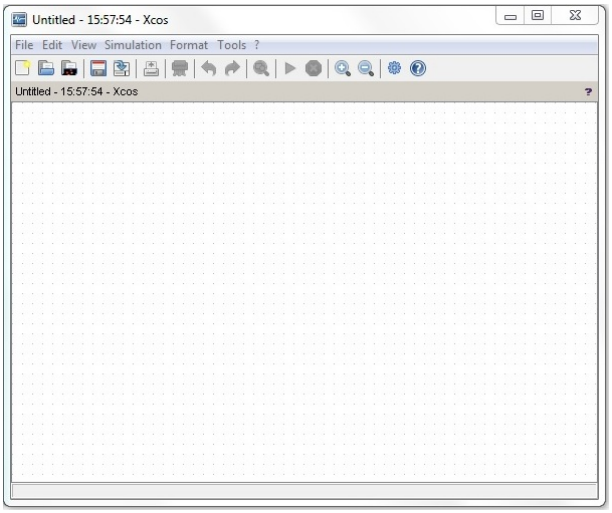
Na obrázku 18 můžeme vidět paletu všech použitelných bloků v programu Xcos. Bloky jsou rozděleny do kategorií. Pro větší přehlednost si můžeme vytvořit vlastní složku

a do ní si vložit používané kategorie, nebo si můžeme po kliknutí na složku palette vybrat jaké kategorie chceme zobrazovat. Kategorie si můžeme i přejmenovávat. Menu složku palette můžeme vidět na obrázku 19.



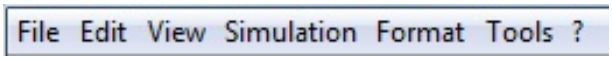
Obrázek 19: Menu složky palette

1.9 Popis pracovní plochy Xcos



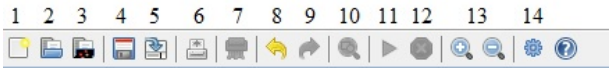
Obrázek 20: Xcos editor

Na obrázku 20 vidíme editor Xcos. Do editoru vkládáme přetažením bloky z palety. Chybové hlášení se nám zapisují do konzole Scilab.



Obrázek 21: Menu Xcos

Na obrázku 21 můžeme vidět všechna menu pro Xcos. V prvním menu file nalezneme stejně jako v programu Scilab tlačítka pro práci se soubory. Například otevření souboru, uložení souboru, exportování a možnosti tisku. Menu edit slouží pro práci s bloky, nalezneme zde tlačítka jako zpět, vpřed, kopírovat, vložit a nastavení parametrů k bloku. V menu view si můžeme přibližovat a oddalovat pracovní plochu, spustit paletu bloků. Menu simulation nám slouží pro nastavení parametru simulace, kompilace, spuštění a vypnutí signalizace. V menu format si můžeme upravovat bloky pomoci tlačítek otočit, přetočit, zrcadlit, také zde můžeme bloky vyrovnávat dle potřeby. Dále si zde můžeme změnit barvu pracovní plochy a zapnout nebo vypnout mřížku.
V této kapitole jsem čerpal z literatury [1] [2]



Obrázek 22: Tlačítka Xcos

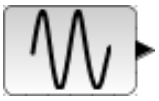
1- Tlačítko spuštění nového diagramu
2- Tlačítko pro otevření souboru
3- Tlačítka pro otevření souboru ze složky Scilab
4- Tlačítko uložit
5- Tlačítko uložit jako
6- Tlačítko tisk
7- Tlačítko smazání
8- Tlačítko zpět
9- Tlačítko vpřed
10- Tlačítko pro přizpůsobení bloku do okna
11- Tlačítko spuštění simulace
12- Tlačítko zastavení simulace
13- Tlačítka pro přiblížení a oddálení pracovní plochy
14- Tlačítka demonstračních ukázek

2 Detailní popis dílčích části "xcos"souboru na jednoduchých modelech

V této kapitole si řekneme něco o tom jak vypadají grafické bloky z programu Xcos v textové podobě. Ukážeme si to na příkladech kde budeme mít jeden blok, ve kterém přesně uvidíme jak se blok zapisuje do textové podoby. V dalším příkladu budeme mít dva bloky a jeden spoj, kde si hlavně zaměříme jak se zapisuje do textové podoby celkový spoj mezi dvěma bloky. V posledním příkladu budeme mít tři bloky a jeden uzel, kde si hlavně ukážeme jak se uzel zapisuje do textové podoby. Dále si ukážeme celkovou strukturu programu.

2.1 Detailní rozbor souboru .xcos obsahující jeden blok

V této podkapitole si ukážeme detailní rozbor souboru .xcos do kterého jsme si uložili jeden blok, v našem případě byl zvolen generátor sinusového signálu (GENSIN_f), který můžeme vidět na obrázku 23. Soubor .xcos si můžeme otevřít v libovolném textovém prohlížeči. Postupně ve výpisech si budeme ukazovat nejdůležitější části programu, jelikož je soubor .xcos textově obsáhlý. Celý soubor bude k nahlédnutí v přílohách.



Obrázek 23: Blok GENSIN_f

```
<BasicBlock dependsOnT="0" id="7ab663e0:158bb0ed30b:-7f76"
    interfaceFunctionName="GENSIN_f" parent="7ab663e0:158bb0ed30b:-7fa9"
    simulationFunctionName="gensin" simulationFunctionType="DEFAULT" style
    ="GENSIN_f;flip=false;mirror=false">
```

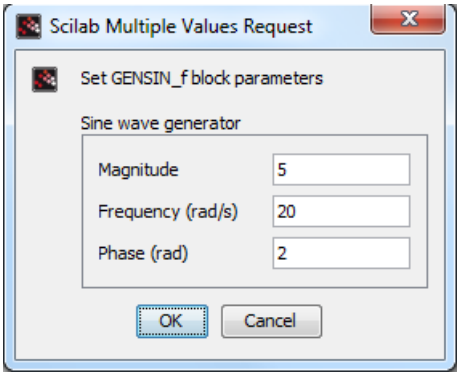
Výpis 1: Začátek programu pro modelování bloku GENSIN_f

Na výpisu 1 můžeme vidět začátek programu pro modelování bloku generátoru sinusového signálu, který začíná <BasicBlock>. Dále následuje id samotného bloku, které je důležité pro následnou komunikaci mezi dalšími bloky. InterfaceFunctionName udává pouze jméno funkce. Pomocí parenta blok komunikuje s blokem, který nese zadané id. SimulationFunctionName opět udává pouze jméno simulační funkce. SimulationFunctionType slouží pro zadání typu simulační funkce. V poslední řadě nám zbývá flip=false a mirror=false. Flip a mirror může nabývat hodnot false nebo true, flip slouží pro překlopení bloku, pokud je false blok překlopen není. Mirror slouží pro zrcadlení bloku, opět když nabývá hodnoty false blok zrcadlen není.

```
<ScilabString as="exprs" height="3" width="1"><data column="0" line="0"
    value="5"/><data column="0" line="1" value="20"/><data column="0" line
    ="2" value="2"/></ScilabString><ScilabDouble as="realParameters"
    height="3" width="1"><data column="0" line="0" realPart="5.0"/><data
    column="0" line="1" realPart="20.0"/><data column="0" line="2"
    realPart="2.0"/></ScilabDouble><ScilabDouble as="integerParameters"
    height="0" width="0"/><Array as="objectsParameters" scilabClass="
    ScilabList"/><ScilabDouble as="nbZerosCrossing" height="1" width="1"><
    data column="0" line="0" realPart="0.0"/></ScilabDouble><ScilabDouble
    as="nmode" height="1" width="1"><data column="0" line="0" realPart="
    0.0"/></ScilabDouble><Array as="oDState" scilabClass="ScilabList"/><
    Array as="equations" scilabClass="ScilabList"/>
```

Výpis 2: Zadávaní dat do bloku GENSIN_f

Hned z počátku můžeme vidět jak se vytváří tabulka do které zadáváme hodnoty. Jak můžeme vidět z příkazu height="3"width="1 tabulka bude mít jedno pole na šířku a 3 pole na výšku. Dale můžeme vidět jak se pomoci příkazu <data> zapisují hodnoty do vytvořené tabulky. Příkaz <data> obsahuje column a line pro orientaci kam se hodnota vloží a neposlední řadě value což je samotná hodnota. Vytvořenou tabulku z programu Xcos můžeme vidět na obrázku 24. Dále z výpisu můžeme vidět, že se hodnoty zapisují ještě jednou a to ve formě reálných čísel pro práci s desetinnými čísly.



Obrázek 24: Tabulka dat bloku GENSIN_f

```
<mxGeometry as="geometry" height="40.0" width="60.0" x="100.0" y="90.0"/>
```

Výpis 3: Umístění bloku GENSIN_f

Na výpisu 3 můžeme vidět jak se umisťuje blok a jak se zadává jeho velikost. Pro tuto funkci nám slouží příkaz <mxGeometry>, který má proměnné height a width, které nám udávají velikost a šířku bloku. Dále zadáváme X a Y což nám určuje kde bude blok umístěn na pracovní ploše programu Xcos.

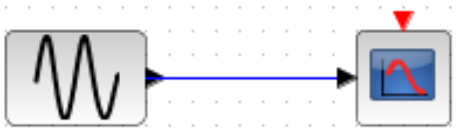
```
<ExplicitOutputPort dataColumns="1" dataLines="1" dataType="REAL_MATRIX"
    id="-269db79c:158ede48e77:-7fb8" ordering="1" parent="7ab663e0:158
    bb0ed30b:-7f76" style="ExplicitOutputPort;align=right;verticalAlign=
    middle;spacing=10.0;rotation=0;flip=false;mirror=false" value=""><
    mxGeometry as="geometry" height="8.0" width="8.0" x="60.0" y="16.0"
    /></ExplicitOutputPort>
```

Výpis 4: Detailní rozbor výstupního portu

Výstupní port je v programu Xcos udělán podobně jako celý blok, v Xcosu je vyobrazen jako černý trojúhelník, jeho konfiguraci můžeme vidět ve výpisu 4. Jak můžeme vidět výstupní port má stejně jako blok id a parenta, kterým je vždy blok ke kterému patří. Dále zde můžeme vidět nastavení funkcí jako zarovnání, mezery, otáčení, překlopení, zrcadlení. Opět zde máme příkaz <mxGeometry> který stejně jako u bloku slouží pro udávaní velikosti a umístění na pracovní ploše Xcosu.

2.2 Detailní rozbor souboru .xcos obsahující dva bloky

V minulé podkapitole jsme si ukázali jak je tvořen soubor .xcos s jedním blokem. V této podkapitole si hlavně ukážeme tvorbu druhého bloku a spoj mezi bloky. Jako první blok jsme nechali generátor sinusového signálu (GENSIN_f) a jako druhý blok použijeme osciloskop (CSCOPE) a poté je spojíme jedním propojem bez uzlu. Zapojení můžeme vidět na obrázku 25.



Obrázek 25: Blok GENSIN_f a CSCOPE se spojem

```
<BasicBlock dependsOnU="1" id="7ab663e0:158bb0ed30b:-7f66"
    interfaceFunctionName="CSCOPE" parent="7ab663e0:158bb0ed30b:-7fa9"
    simulationFunctionName="cscope" simulationFunctionType="C_OR_FORTRAN"
    style="CSCOPE;flip=false;mirror=false">
```

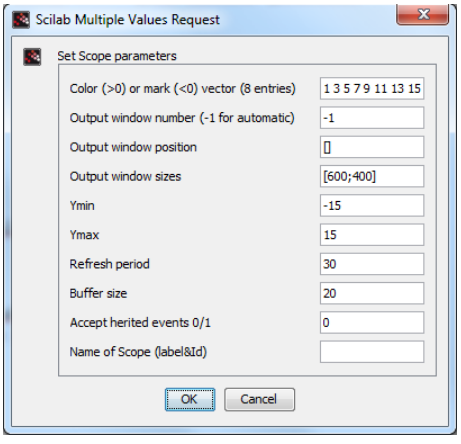
Výpis 5: Začátek programu pro modelování bloku cscope

Z výpisu 5 můžeme vidět začátek programu pro modelování bloku osciloskopu, který začíná stejně jako generátor sinusového signálu příkazem <BasicBlock>. Dále následuje id samotného bloku, které je jiné než u generátoru sinusového signálu, z toho tedy vyplývá že id pro každý blok je jiné. InterfaceFunctionName udává jméno funkce tedy CSCOPE. Pomocí parenta blok komunikuje s blokem, který nese zadané id. SimulationFunctionName opět udává jméno simulační funkce tedy cscope. Stejně jako u generátoru sinusového signálu zde máme funkce flip a mirror.

```
<ScilabString as="exprs" height="10" width="1"><data column="0" line="0"
    value="1 3 5 7 9 11 13 15"/><data column="0" line="1" value="-1"/><
    data column="0" line="2" value="[]"/><data column="0" line="3" value="
    [600;400]"/><data column="0" line="4" value="-15"/><data column="0"
    line="5" value="15"/><data column="0" line="6" value="30"/><data
    column="0" line="7" value="20"/><data column="0" line="8" value="0"/><
    data column="0" line="9" value=""/></ScilabString><ScilabDouble as="
    realParameters" height="4" width="1"><data column="0" line="0"
    realPart="0.0"/><data column="0" line="1" realPart="-15.0"/><data
    column="0" line="2" realPart="15.0"/><data column="0" line="3"
    realPart="30.0"/></ScilabDouble><ScilabDouble as="integerParameters"
    height="15" width="1"><data column="0" line="0" realPart="-1.0"/><data
     column="0" line="1" realPart="1.0"/><data column="0" line="2"
    realPart="20.0"/><data column="0" line="3" realPart="1.0"/><data
    column="0" line="4" realPart="3.0"/><data column="0" line="5" realPart
    ="5.0"/><data column="0" line="6" realPart="7.0"/><data column="0"
    line="7" realPart="9.0"/><data column="0" line="8" realPart="11.0"/><
    data column="0" line="9" realPart="13.0"/><data column="0" line="10"
    realPart="15.0"/><data column="0" line="11" realPart="-1.0"/><data
    column="0" line="12" realPart="-1.0"/><data column="0" line="13"
    realPart="600.0"/><data column="0" line="14" realPart="400.0"/></
    ScilabDouble><Array as="objectsParameters" scilabClass="ScilabList"/><
    ScilabDouble as="nbZerosCrossing" height="1" width="1"><data column="0
    " line="0" realPart="0.0"/></ScilabDouble>
```

Výpis 6: Zadávaní dat do bloku cscope

Stejně jako v minulé podkapitole se na začátku vytvoří tabulka height="10"width="1"což nám naznačuje, že tabulka bude mít jedno pole na šířku a 10 polí na výšku. Poté se zapisují hodnoty pomocí řádků a sloupců stejně jako jsme popisovali u generátoru sinusového signálu. Opět se zapisují hodnoty dvakrát, podruhé jako reálná čísla pro práci s desetinnými čísly. Tabulku naplněnou hodnotami z programu Xcos můžeme vidět na obrázku 26.



Obrázek 26: Tabulka dat bloku CSCOPE

```
<mxGeometry as="geometry" height="40.0" width="40.0" x="250.0" y="90.0"/>
```

Výpis 7: Umístění bloku cscope

Stejně jako u generátoru sinusového signálu nám pro umístění a změnu velikosti slouží příkaz <mxGeometry>. pomocí "height"a "width"nastavujeme výšku a šířku bloku a pomoci X a Y posouváme po pracovní ploše programu Xcos.

```
<ExplicitInputPort dataColumns="1" dataType="REAL_MATRIX" id="7ab663e0:158
    bb0ed30b:-7f65" ordering="1" parent="7ab663e0:158bb0ed30b:-7f66" style
    ="ExplicitInputPort;align=left;verticalAlign=middle;spacing=10.0;
    rotation=0;flip=false;mirror=false" value=""><mxGeometry as="geometry"
     height="8.0" width="8.0" x="-8.0" y="16.0"/>
```

Výpis 8: Detailní rozbor vstupního portu cscope

Vstupní port je podobný výstupnímu portu liší se v grafickém zobrazení. Vstupní port má vrchol trojúhelníku směřující do bloku a výstupní port má vrchol směřující od bloku do prostoru.Z výpisu 8 můžeme vidět detailní rozbor vstupního portu. Stejně jako u výstupního portu zde máme id a parenta, který nese id osciloskopu. Dále zde vidíme nastavení funkcí pro seřazení, zarovnání, mezery, zrcadlení, otáčení. V poslední řadě zde máme příkaz <mxGeometry>, který udává velikost a pozici portu.

```
<ControlPort dataType="" id="7ab663e0:158bb0ed30b:-7f64" ordering="1"
    parent="7ab663e0:158bb0ed30b:-7f66" style="ControlPort;align=center;
    verticalAlign=top;spacing=10.0;rotation=90;flip=false;mirror=false"><
    mxGeometry as="geometry" height="8.0" width="8.0" x="16.0" y="-8.0"
    /></ControlPort>
```

Výpis 9: Detailní rozbor kontrolního portu cscope

Kontrolní port je podobný jako vstupní či výstupní port jen je zbarven červeně a vrchol vstupuje do bloku. Když se podíváme na jeho textovou strukturu tak obsahuje také své id a parenta osciloskopu, dále zde vyčteme nastavení funkcí pro seřazení, zarovnání, mezery, zrcadlení, otáčení a v poslední řadě zde máme příkaz <mxGeometry>, ze kterého vyčteme rozměry a umístění bloku na pracovní ploše.

```
<ExplicitLink id="305999f9:15912724dc7:-7e6e" parent="7ab663e0:158bb0ed30b
    :-7fa9" source="7ab663e0:158bb0ed30b:-7f75" target="7ab663e0:158
    bb0ed30b:-7f65"><mxGeometry as="geometry"><mxPoint as="sourcePoint" x=
    "250.0" y="110.0"/><mxPoint as="targetPoint" x="160.0" y="110.0"/></
    mxGeometry></ExplicitLink>
```

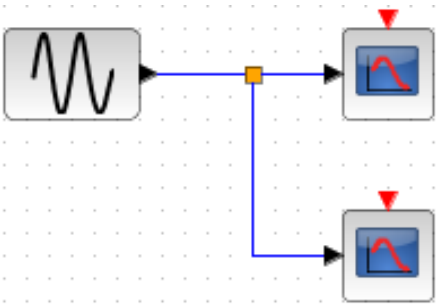
Výpis 10: Detailní rozbor spoje mezi dvěma bloky

Z výpisu 10 můžeme vidět, že spoj má své id a parenta odkazujícího na defaultní id. Dále vidíme id zdroje(source) a id cíle(target), které jsou v našem případě pro source id

výstupního portu generátoru sinusového signálu a pro target id vstupního portu osciloskopu. Z těchto id vidíme odkud(source) kam(target) je spoj veden. Dále zde máme příkaz <mxGeometry> ve kterém vidíme x-ové a y-ové souřadnice zdroje a cíle spoje.

2.3 Detailní rozbor souboru .xcos obsahující tři bloky a uzel

V minulých podkapitolách jsme si ukázali jak se tvoří celý blok, vstupní, výstupní a kontrolní porty a na konec jsme si ukázali jak se tvoří spoj mezi dvěma bloky. V této podkapitole si ukážeme na jednoduchém zapojení tří bloků(generátor sinusového signálu a dva osciloskopy) jak se tvoří uzel, který spojuje tyto tři bloky. Toto jednoduché zapojení můžeme vidět na obrázku 27.



Obrázek 27: Zapojení tří bloků uzlem

```
<SplitBlock id="305999f9:15912724dc7:-7f53" parent="7ab663e0:158bb0ed30b
    :-7fa9" simulationFunctionType="DEFAULT" style="SPLIT_f;flip=false;
    mirror=false"><mxGeometry as="geometry" height="7.0" width="7.0" x="
    207.0" y="107.0"/></SplitBlock>
```

Výpis 11: Tvorba modelu uzlu

Tvorba modelu uzlu je podobná tvorbě portů. Opět zde máme id a parenta v ve většině případů odkazujícího na id pracovní plochy. Dále se zde nastavuje typ simulační funkce, styl, otočení a zrcadlení. Nakonec zde máme příkaz <mxGeometry> pomoci kterého se nastavuje velikost uzlu a souřadnice uzlu.

```
<ExplicitInputPort dataType="UNKNOW_TYPE" id="305999f9:15912724dc7:-7f52"
    ordering="1" parent="305999f9:15912724dc7:-7f53" style="
    ExplicitInputPort;align=left;verticalAlign=middle;spacing=10.0;
    rotation=0;flip=false;mirror=false" visible="0"><mxGeometry as="
    geometry" height="8.0" width="8.0" x="-8.0" y="-4.0"/></
    ExplicitInputPort>
<ExplicitOutputPort dataType="UNKNOW_TYPE" id="305999f9:15912724dc7:-7f51"
     ordering="1" parent="305999f9:15912724dc7:-7f53" style="
    ExplicitOutputPort;align=right;verticalAlign=middle;spacing=10.0;
    rotation=0;flip=false;mirror=false" visible="0"><mxGeometry as="
```

```
    geometry" height="8.0" width="8.0" x="7.0" y="-4.0"/></
    ExplicitOutputPort>
<ExplicitOutputPort dataType="UNKNOW_TYPE" id="305999f9:15912724dc7:-7f50"
     ordering="2" parent="305999f9:15912724dc7:-7f53" style="
    ExplicitOutputPort;align=right;verticalAlign=middle;spacing=10.0;
    rotation=0;flip=false;mirror=false" visible="0"><mxGeometry as="
    geometry" height="8.0" width="8.0" x="7.0" y="-4.0"/></
    ExplicitOutputPort>
```

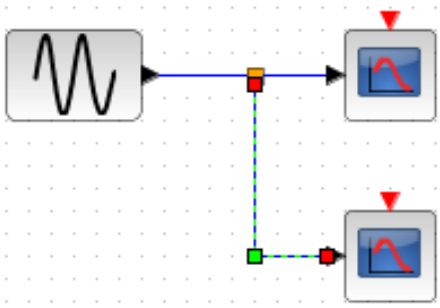
Výpis 12: Tvorba vstupních a výstupních portů uzlu

V minulém výpisu jsme si ukázali jak se tvoří uzel jako blok, teď se podíváme jak se vytváří vstupní a výstupní porty na uzlu, které můžeme vidět ve výpisu 12. Tyto porty se vytváří stejně jako porty u bloků. Nejdříve se vytvoří id a parent odkazující na id modelu uzlu, přidá se styl zda bude port fungovat jako vstup nebo výstup. Poté se nastavuje zarovnání, mezery, otočení, zrcadlení a viditelnost. V poslední řadě zde máme <mxGeometry> pro nastavení velikosti portu a nastavení pozice na pracovní ploše.

```
<ExplicitLink id="305999f9:15912724dc7:-7f4f" parent="7ab663e0:158bb0ed30b
    :-7fa9" source="7ab663e0:158bb0ed30b:-7f75" target="305999f9:15912724
    dc7:-7f52"><mxGeometry as="geometry"><Array as="points" scilabClass="
    ScilabList"/></mxGeometry></ExplicitLink>
<ExplicitLink id="305999f9:15912724dc7:-7f4e" parent="7ab663e0:158bb0ed30b
    :-7fa9" source="305999f9:15912724dc7:-7f51" target="7ab663e0:158
    bb0ed30b:-7f65"><mxGeometry as="geometry"><Array as="points"
    scilabClass="ScilabList"/></mxGeometry></ExplicitLink>
<ExplicitLink id="305999f9:15912724dc7:-7f54" parent="7ab663e0:158bb0ed30b
    :-7fa9" source="305999f9:15912724dc7:-7f50" target="7ab663e0:158
    bb0ed30b:-7f5d"><mxGeometry as="geometry"><mxPoint as="sourcePoint" x=
    "210.0" y="110.0"/><mxPoint as="targetPoint" x="240.0" y="180.0"/><
    Array as="points" scilabClass="ScilabList"><mxPoint x="210.0" y="190.0
    "/></Array></mxGeometry></ExplicitLink>
```

Výpis 13: Tvorba spojů mezi bloky a uzlem

Teď když máme vytvořený uzel a jeho vstupní a výstupní porty stačí nám bloky mezi sebou spojit. z výpisu 13 můžeme vidět pro první dva spoje, že jsou podobné jako spoj mezi dvěma bloky tedy klasicky id a parent odkazující na uzel. Pro první spoj je id source a id target veden mezi výstupním portem generátoru sinusového signálu a vstupním portem uzlu, pro druhý spoj je id source a id target veden mezi výstupním portem uzlu a vstupním portem osciloskopu. Třetí spoj je veden mezi uzlem a druhým spodním osciloskopem, ale spoj je udělán trochu jinak, je tam přidán bod, který můžeme vidět na obrázku 28.



Obrázek 28: Vyobrazení bodu na spoji

Bod je vyobrazen zeleným čtvercem v prostředí Xcos. Díky tohoto bodu můžeme podle potřeby posouvat spoj mezi bloky. Když se podíváme zpět na výpis 13, tak v textové podobě je začátek pro tento spoj stejný, tedy má vlastní id a parent odkazující na uzel, id source a id target je veden mezi výstupním portem uzlu a vstupním portem spodního osciloskopu. <mxGeometry> je v tomto případě obohacen o souřadnice "sourcePoint"tedy souřadnice výstupního portu uzlu a souřadnice "targetPoint"tedy souřadnice vstupního portu spodního osciloskopu. V poslední řadě tu máme příkaz <mxPoint> kam se nastavili souřadnice samotného bodu.
V této kapitole jsem čerpal z literatury [1] [3]

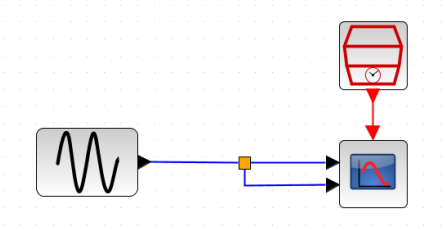
3 Tvorba modelů pomocí jazyka Scilab

Simulink	Xcos
Gain	GAINBLK_f
Add	BIGSOM_f
Sine Wave	GENSIN_f
Time Scope	CMSCOPE

Tabulka 1: Převod bloků ze Simulinku do Scilabu

3.1 Popis tvorby jednoduchého zapojení s uzlem

V této kapitole si ukážeme vytvoření jednoduchého diagramu za pomoci jazyka Scilab, který se následně zobrazí v okně Xcos. Jako příklad si ukážeme diagram, který bude obsahovat dva dobře známé bloky z minulých kapitol, tedy generátor sinusového signálu a osciloskop, poté budou bloky spojeny uzlem. Dále bude zapojení obsahovat blok časového signálu (SampleCLK) připojený k osciloskopu. Zapojení můžeme vidět na obrázku 29.



Obrázek 29: Blokové zapojení

Než začneme vůbec bloky nebo diagram vytvářet musíme si načíst Xcos knihovnu a to pomoci *loadXcosLibs*. Nyní když máme knihovnu načtenou můžeme se zaměřit na vytváření diagramu, do kterého budeme bloky načítat. Tvorbu diagramu vidíme ve výpisu 14.

```
loadXcosLibs();                          //nacteni xcos knihovny

scs_m=scicos_diagram(..                  // vytvoreni diagramu
    version="scicos4.2",..               // verze
    props=scicos_params(..               // parametry diagramu
        wpar=[600,450,0,0,600,450],..// rozmery pracovni plochy
        Title=["SciNotesToXcos"],..  // nazev diagramu
        tol =[0.0001;0.000001;1.000E-10;100010;0;0;0],..
        // nastaveni parametru simulace
        tf=80,..                         // cas konecneho zacleneni
```

```
    options=tlist(["scsopt","3D","Background","Link","ID","Cmap"],list
        (%t,33),[8,1],[1,5],..   // nastaveni
```

Výpis 14: Tvorba diagramu

Jak můžeme vidět z výpisu 14, k tvorbě diagramu slouží příkaz `scicos_diagram` ve kterém se určuje pouze verze scicosu. Dále si určujeme parametry diagramu pomoci příkazu `scicos_params`. V parametrech diagramu máme proměnnou "tol",která nám určuje nastavení parametrů simulace a to v pořadí: Absolutní tolerance integrátoru;Relativní tolerance integrátoru; Tolerance na čas; Maximální časový interval začlenění; škálování v reálném čase; Druh řešitele(nabývá hodnot 0-7); Maximální velikost kroku(0 znamená bez limitu) a v poslední řadě proměnná "tf"což je čas konečného začlenění. Všechny tyto hodnoty najdeme v tabulce Xcosu(Simulace->Nastavení). Nyní se podíváme jak se vytváří grafické nastavení bloku.

```
scs_m.objs(1)=scicos_block(.. // vytvoreni bloku
    gui="GENSIN_f",..    // nazev funkce
    graphics=scicos_graphics(.. // graficke nastaveni bloku
        orig=[50,-80],.. // pozice bloku
        sz=[60,40],..    // velikost bloku
        flip=%t,.. // orientace bloku
        theta=0,.. // zmena uhlu
        exprs=["1";"1";"0"],.. // zadavani hodnot do boxu
        pin=[],..   // pocet spoju pripojenych na vstupni port
        pout=[4],.. // pocet spoju pripojenych na vystupni port
        pein=[],..  // pocet spoju pripojenych na vstupni port pro
            udalost
        peout=[],.. // pocet spoju pripojenych na vystupni port pro
            udalost
        gr_i=list("xstringb(orig(1),orig(2),[''GENSIN_f''],sz(1),sz(2)
            ,''fill'');",8),..
        // graficke instrukce
        id="",..    // identifikacni popis
        in_implicit=[],..       // typ vstupniho portu
        out_implicit=["E"]),..  // typ vystupniho portu
```

Výpis 15: Tvorba bloku a graficke nastavení

Z výpisu 15 můžeme vidět jak se tvoří blok generátoru sinusového signálu a nastavuje jeho grafická struktura. Můžeme se zaměřit na poslední dva řádky, kde máme typ vstupního a výstupního portu. Tyto proměnné nabývají hodnoty "E"a "I", kde "E"znamená explicitní port a "I"implicitní port. Nyní se podíváme jak se tvoří modelová část bloku generátoru sinusového signálu.

```
    model=scicos_model(.. // Vytvoreni modelu
        sim="gensin",.. // Nazev/typ vypocetni funkce
        in=[],..     // Prvni rozmer vstupnich portu
        in2=[],..    // Druhy rozmer vstupnich portu
        intyp=[],..  // Datovy typ vstupnich portu
        out=[1],..   // Prvni rozmer vystupnich portu
```

```
        out2=[1],..  // Druhy rozmer vystupnich portu
        outtyp=[1],..// Datovy typ vystupnich portu
        evtin=[],..  // Velikost vstupnich portu pro udalost
        evtout=[],.. // Velikost vystupnich portu pro udalost
        state=[],..  // Pocatecni stav prubezneho stavu
        dstate=[],.. // Pocatecni stav diskretniho stavu
        odstate=[],..// Pocatecni stav objektu v diskretnim stavu
        rpar=[1;1;0],.. // Celociselne parametry
        ipar=[],..   // Parametry s plovouci desetinou carkou
        opar=[],..   // Parametry obejktu
        blocktype="c",..// Typ bloku
        firing=%f,.. // Prvotni cas vystupni usalosti
        dep_ut=[%t,%f],.. // Vlastnosti
        label="",..  // Popis bloku
        nzcross=0,.. // Pocet pruchodu nulou
        nmode=0,..   // Pocet rezimu
```

Výpis 16: Modelové nastavení bloku

Z výpisu 16 vidíme jak vypadá celé modelové nastavení bloku generátoru sinusového signálu. Místa kde jsou prázdné hranaté závorky znamenají, že blok tyto proměnné nevyužívá, nebo nabývají nulové hodnoty. Nyní si naprosto stejným způsobem vytvoříme osciloskop a blok časového signálu SampleCLK. Uzel se tvoří velmi podobně jako bloky generátoru a osciloskopu. Pojd'me se tedy podívat na to jak se tvoří blok uzlu.

```
scs_m.objs(5)=scicos_block(..   // Vytvoreni bloku
    gui="SPLIT_f",.. // Nazev funkce
    graphics=scicos_graphics(.. // Graficke nastaveni bloku
        orig=[170,-64],..       // Pozice bloku
        sz=[7,7],..  // Velikost bloku
        flip=%t,..   // Orientace bloku
        theta=0,..   // Zmena uhlu
        exprs=[],..  // Zadavani hodnot do boxu
        pin=4,..     // Pocet spoju pripojenych na vstupni port
        pout=[5;6],..// Pocet spoju pripojenych na vystupni port
        pein=[],..   // Pocet spoju pripojenych na vstupni port pro
            udalost
        peout=[],..  // Pocet spoju pripojenych na vystupni port pro
            udalost
        gr_i=list("xstringb(orig(1),orig(2),''SPLIT_f'',sz(1),sz(2),''
            fill'')",8),..    // Graficke instrukce
        id="",..     // Identifikacni popis
        in_implicit=["E"],..      // Typ vstupniho portu
        out_implicit=["E";"E"]),..// Typ vystupniho portu
```

Výpis 17: Vytvoření a graficke nastavení bloku uzlu

Jak můžeme vidět z výpisu 17 vytvoření a grafické nastavení je velice podobné jako nastavení ostatních bloků, funkce uzlu se nazývá "SPLIT_f". Jak můžeme vidět z prvního řádku jedná se o už pátý blok nebo spoj v tomto digramu (objs(5)). Velikost uzlů je menší než velikost důležitých bloků, nabývá ideálně hodnoty [7,7] oproti ostatním blokům, které nabývají hodnot pro čtvercové [40,40] a obdélníkové [60,40] bloky. Nenabývá

žádných hodnot v proměnné 'exprs'. Dále si můžeme opět povšimnout dvou posledních řádků výpisu, kde jsme si ukázali u generátoru že má pouze jeden výstupní port explicitního typu, zde máme jeden vstupní port a dva výstupní porty explicitního typu. Pojďme se tedy podívat dále jak se tvoří modelové nastavení uzlu.

```
model=scicos_model(.. // vytvoreni modelu
    sim="lsplit",.. // nazev/typ vypocetni funkce
    in=[-1],..      // Prvni rozmer vstupnich portu
    in2=[-2],..     // Druhy rozmer vstupnich portu
    intyp=[-1],..   // Datovy typ vstupnich portu
    out=[-1;-1],..  // Prvni rozmer vystupnich portu
    out2=[-2;-2],.. // Druhy rozmer vystupnich portu
    outtyp=[-1;-1],..// Datovy typ vystupnich portu
    evtin=[],..     // Velikost vstupnich portu pro udalost
    evtout=[],..    // Velikost vystupnich portu pro udalost
    state=[],..     // Pocatecni stav prubezneho stavu
    dstate=[],..    // Pocatecni stav diskretniho stavu
    odstate=[],..   // Pocatecni stav objektu v diskretnim stavu.
    rpar=[],..      // Celociselne parametry
    ipar=[],..      // Parametry s plovouci desetinou carkou
    opar=[],..      // Parametry obejktu
    blocktype="c",..// Typ bloku
    firing=[],..    // Prvotni cas vystupni udalosti
    dep_ut=[%t,%f],..// Vlastnosti
    label="",..     // Popis bloku
    nzcross=0,..    // Pocet pruchodu nulou
    nmode=0,..      // Pocet rezimu
```

Výpis 18: Modelové nastavení bloku uzlu

Modelové nastavení uzlu hlavně nastavuje hodnoty vstupních a výstupních portů. Jak můžeme vidět z prvních řádků výpisu 18, nastavujeme zde hodnoty pro první a druhý výstupní a vstupní port. Dále si zde nastavujeme typ vstupních a výstupních portů. Ostatní hodnoty jsou prázdné nebo nulové. Teď když máme vytvořené bloky a uzel, podíváme se na tvorbu jednotlivých spojů mezi nimi.

```
scs_m.objs(3)=scicos_link(.. // vytvoreni spoje
        xx=[50;170],..   // Xove souradnice
        yy=[-80;-64],..  // Yove souradnice
        id="drawlink",.. // identifikacni popis
        thick=[1,1],..   // tloustka spoje
        ct=[1,1],..      // [barva, charakter] spoje
        from=[1,1,0],..  // odkud spoj povede
        to=[5,1,1])      // kam spoj povede
```

Výpis 19: Tvorba spoje mezi bloky

Z výpisu 19 vidíme tvorbu spoje mezi generátorem sinusového signálu a uzlem. Xové a Yové souřadnice nám určují kudy spoj povede. Proměnná "from"nám určuje odkud spoj povede, kde první číslo nám říká ze, kterého bloku spoj povede(v našem případě 1

generátor, 2 osciloskop, 5 uzel), druhé číslo určuje číslo portu (ikdyž máme více vstupních a výstupních portů, začíná číslem 1 pro vstupní i výstupní port) a třetí číslo určuje zda jde o výstupní port nebo o vstupní port, kde 0 je výstupní port a 1 je vstupní port. to stejné platí i pro proměnou "to". V poslední řadě si ukážeme jak se tvoří spoj s bodem, který jsme viděli v minulé kapitole.

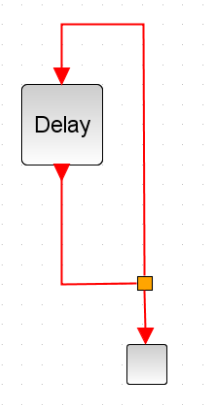
```
scs_m.objs(6)=scicos_link(..
        xx=[170;173;240],..  // Xove souradnice
        yy=[-64;-74;-87],..  // Yove souradnice
        id="drawlink",..     // identifikacni popis
        thick=[1,1],..       // tloustka spoje
        ct=[1,1],..          // [barva, charakter] spoje
        from=[5,1,0],..      // odkud spoj povede
        to=[2,2,1])          // kam spoj povede
```

Výpis 20: Tvorba spoje mezi bloky s bodem

Z výpisu 20 můžeme vidět, že tvorba bodu spočívá v tom že si do Xových a Yových souřadnic přidáme třetí souřadnici, pak už je to naprosto stejné jako u běžného spoje. Z výpisu můžeme vidět, že spoj vede mezi uzlem(5) a druhým portem osciloskopu.

3.2 Tvorba bloku CLOCK_f

V této kapitole si řekneme něco o tvorbě bloku CLOCK_f a jak se liší tvorbou oproti bloku SampleCLK. Blok CLOCK_f je daleko složitější na vytvoření než blok SampleCLK, jelikož při tvorbě bloku CLOCK_f musíme zvlášť vytvořit nový diagram a do diagramu vložit bloky DELAY_f(zpožď'ovač), CLKOUT(výstupní port), uzel a spoje mezi bloky. Vytvořené blokové zapojení můžeme vidět na obrázku 30. Nyní si ukážeme vytvoření nového diagramu a tvorbu bloku DELAY_f, jelikož do tohoto bloku ses zapisují hodnoty, které používáme pro nastavení bloku CLOCK_f.



Obrázek 30: Blokové zapojení pro blok CLOCK_f

```
scs_m_1=scicos_diagram(..           // vytvoreni diagramu
    version="scicos4.2",..          // verze
    props=scicos_params(..          // parametry diagramu
        wpar=[600,450,0,0,600,450],..// rozmery pracovni plochy
        Title="",..                 // nazev diagramu
        tol=[0;0;0;0;0;0],..        // nastaveni parametru simulace
        tf=100000,..                // cas konecneho zacleneni
        options=tlist(["scsopt","3D","Background","Link","ID","Cmap"],list
            (%t,30),8,[1,5],..      // nastaveni
```

Výpis 21: Tvorba nového diagramu pro blok CLOCK_f

Jak můžeme vidět z výpisu 21 nový diagram se moc neliší od hlavního diagramu. Můžeme si povšimnout proměnné "tol", která nenabývá žádné hodnoty jako to je u hlavního programu, pouze nastavujeme čas konečného začlenění. Dále digramu nedáváme žádnou hlavičku. Zbytek nastavení už je totožný s hlavním diagramem. Teď se podíváme na tvorbu bloku DELAY_f.

```
    scs_m_1.objs(3)=scicos_block(..     // vytvoreni bloku
            gui="EVTDLY_f",..// nazev funkce
            graphics=scicos_graphics(.. // graficke nastaveni bloku
            orig=[320,232],..// pozice bloku
            sz=[40,40],..   // velikost bloku
            flip=%t,..      // orientace bloku
            theta=0,..      // zmena uhlu
            exprs=["0.001";"1"],..// zadavani hodnot do boxu
            pin=[],..       // pocet spoju pripojenych na vstupni port
            pout=[],..      // pocet spoju pripojenych na vystupni port
            pein=7,..       // pocet spoju pripojenych na vstupni port pro
                 udalost
            peout=4,..      // pocet spoju pripojenych na vystupni port
                pro udalost
            gr_i=list("xstringb(orig(1),orig(2),''EVTDLY_f'',sz(1),sz(2)
                ,''fill'')",8),..   // graficke instrukce
            id="",..            // identifikacni popis
            in_implicit=[],..   // typ vstupniho portu
            out_implicit=[]),.. // typ vystupniho portu
```

Výpis 22: Tvorba grafického zobrazení bloku DELAY_f

Tvorba grafického zobrazení bloku DELAY_f je opět podobné ostatním blokům, které jsme si již ukazovali. Pouze si zde můžeme poukázat na proměnou "exprs"do které se zapisují hodnoty pro blok CLOCK_f. Modelovou tvorbu si již ukazovat nemusíme, jelikož je podobná ostatním blokům jen s jinými hodnotami. Teď když máme diagram hotový, můžeme se podívat na tvorbu bloku CLOCK_f.

```
        model=scicos_model(..// vytvoreni modelu
            sim="csuper",.. // nazev/typ vypocetni funkce
            in=[],..        // Prvni rozmer vstupnich portu
            in2=[],..       // Druhy rozmer vstupnich portu
```

```
intyp=1,..      // Datovy typ vstupnich portu
out=[],..       // Prvni rozmer vystupnich portu
out2=[],..      // Druhy rozmer vystupnich portu
outtyp=1,..     // Datovy typ vystupnich portu
evtin=[],..     // Velikost vstupnich portu pro udalost
evtout=1,..     // Velikost vystupnich portu pro udalost
state=[],..     // Pocatecni stav prubezneho stavu
dstate=" ",..   // Pocatecni stav diskretniho stavu
odstate=list(),..// Pocatecni stav objektu v diskretnim stavu
rpar=scs_m_1,.. // Celociselne parametry
ipar=[],..      // Parametry s plovouci desetinou carkou
opar=list(),..  // Parametry obejktu
blocktype="h",..// Typ bloku
firing=%f,..    // Prvotni cas vystupni udalosti
dep_ut=[%f,%f],..// Vlastnosti
label="",..     // Popis bloku
nzcross=0,..    // Pocet pruchodu nulou
nmode=0,..      // Pocet rezimu
```

Výpis 23: Tvorba modelové části bloku CLOCK_f

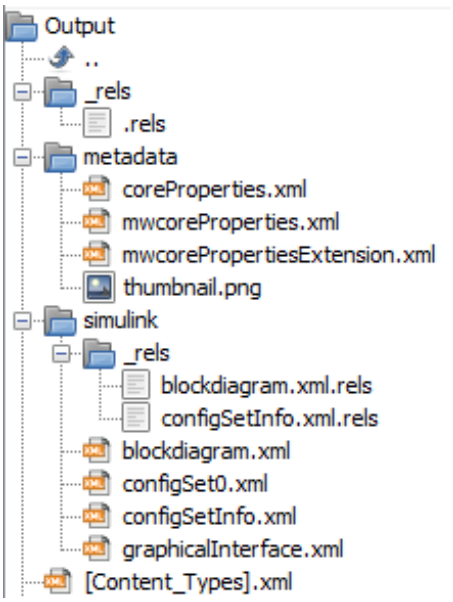
Tvorbu grafického zobrazení si ukazovat nemusíme, jelikož zde není nic jiného, pouze proměnná "exprs"zůstane prázdná. Ve výpisu 23 vidíme tvorbu modelové části, kde si můžeme povšimnout přiřazení nového diagramu do proměnné "rpar".
V této kapitole jsem čerpal z literatury [3]

4 Automatické generování bloků z programu Simulink do programu Scilab

V této kapitole si řekneme něco o načtení a rozbalení souboru Simulink(.slx). V další části se podíváme na stěžejní části načtených bloků a na jejich struktury načtené ze souboru Simulink. Dále se podíváme na načtení spojů a jejich úpravu odpovídající hodnotám potřebným pro Scilab. Nakonci si porovnáme výstup programu se zadaným zapojením v Simulink.

4.1 Rozbalení a načtení souboru Simulink (.slx)

V této kapitole si řekneme jak se rozbaluje a načítá Simulink soubor .slx a co se provede jako první v programu. Program funguje jak pro operační systém windows tak pro linux. Jakmile program spustíme otevře se nám okno, kde zadáme cestu k Simulink souboru .slx. V dalším kroku se otevře okno, kde zadáme cestu složky kam se soubor rozbalí a uloží. Stromové zobrazení rozbaleného souboru .slx můžeme vidět na obrázku 31.



Obrázek 31: Soubory po rozbalení souboru .slx

Program pro windows využívá program třetí strany, v našem případě program 7z. Poté nám ve složce vzniknou tři složky a to _rels, metadata a Simulink. Právě složka Simulink obsahuje soubor "blockdiagram"kam se zapisují všechny informace o nastavení Simulink souboru a informace o blocích se, kterými budeme pracovat. Soubor blockdiagram je zapsán ve struktuře XML, toho můžeme využít pro orientaci a soubor načíst do Scilabu pomoci příkazu *xmlDump(BlockDiagram)*. Dále se můžeme procházet soubor pomoci rootu a childrenu, jelikož víme že soubor je ve struktuře XML a toho jsme také využili při tvorbě našich struktur v jazyce Scilab.

4.2 Generování bloku GAINBLK_f

V této kapitole si ukážeme jak se načítají hodnoty do naši struktury a poté se zapisují do souboru. Načtení bloku gain je tvořeno jedním cyklem for, kde si zjistíme počet proměnných pro tento blok a to pomoci příkazu *size*. Jeden cyklus for načte název atributu pomoci childrenu a příkazu *attributes*, o kterých jsme si řekli v minulé kapitole a hodnotu atributu pomoci příkazu *content*. Poté se nám názvy atributů a hodnoty zapíší do struktury, kterou můžeme vidět na obrázku 32.

```
BlockType: "Gain"
Name: "Gain"
SID: "36"
Gain: "25"
Position: "[215, 35, 245, 65]"
ZOrder: "34"
ParamMin: "0"
ParamMax: "50"
ParamDataTypeStr: "int8"
OutMin: "10"
OutMax: "25"
OutDataTypeStr: "int8"
LockScale: "on"
RndMeth: "Nearest"
SaturateOnIntegerOverflo: "off"
obj: 1
```

Obrázek 32: Struktura Gain

Ještě než se dostaneme k druhému cyklu for pro uložení hodnot do souboru, zjistíme si pomoci příkazu *if exists('Gain') == 1* jestli vůbec tento blok existuje. Pokud neexistuje nastaví se nám do pomocné proměnné číslo 0, pomocná proměna slouží pro nastavení čísla objektu. Pokud existuje tak se provede cyklus for kde se první zjistí kolik bloků gain je v souboru a tolikrát se cyklus provede. Blok gain je jeden z jednodušších bloků, jelikož má pouze jeden neměnný výstupní a vstupní port, tedy zde pracujeme jen z pozicí bloku, názvem bloku a hodnotou bloku. Pozice bloku se musí rozdělit z typu string na číselné vektory, to můžeme vidět ve výpisu 24. Takto jsou tvořené pozice pro všechny ostatní bloky. Do pomocné struktury si ukládáme SID a číslo objektu pro spoje, ale k tomu se dostaneme v další kapitole.

```
pos = Gain(kk).Position     //ulozeni strigu s pozicema
ind = strindex(pos, ',')    //vyhledani pozic carek
pos_1 = strtod(part(pos, 2:ind(1)-1))       //Xsove cislo pozice
pos_2 = strtod(part(pos, ind(1)+1:ind(2)-1))//Ynove cislo pozice
```

Výpis 24: Rozdělení pozic

4.3 Generování bloku BIGSOM_f

Blok add se načítá stejně jako blok gain, tedy jeden cyklus for pro načtení, jen s rozdílem toho že si zde musíme ošetřit vstup. Když si do Simulinku vložíme blok a neupravíme jeho počet portu a necháme nastavené '++', což je základní nastavení, neuloží se nám toto nastavení do souboru blockdiagram. Musíme si tedy ručně přidat tuto možnost do načtení struktury. Vytvořenou strukturu pro blok add můžeme vidět na obrázku 33.

```
BlockType: "Sum"
Name: "Add"
SID: "6"
Inputs: "+-"
Ports: "[2, 1]"
Position: "[290, 151, 320, 184]"
ZOrder: "6"
InputSameDT: "off"
OutDataTypeStr: "Inherit: Inherit via internal rule"
SaturateOnIntegerOverflo: "off"
obj: 5
```

Obrázek 33: Struktura Sum

Stejně jako v případě bloku gain se zde zjistí zda blok add existuje pomoci *if exists('Sum') == 1*, pokud neexistuje tak se do pomocné proměnné uloží číslo objektů gain. Pokud existuje provede se cyklus for, kde se uloží pozice stejné jako vidíme u bloku gain. Dále si uděláme cyklus for pro vstup, jelikož vstup pro Simulink je v plusech a mínusech, ale pro Scilab je plus jednička a mínusko minus jednička. Vyřešení vstupu můžeme vidět z výpisu 25.

```
for ii = 1:N_Inputs     // cyklus pro pocet vstupu
    if strcmp(Inputs(ii, 1), '+') == 0 then
    // pokud je 0 tak se zmeni + na 1
        sciInputs = 1
    else
        sciInputs = -1 // jinak - na -1
    end
    if ii == 1
        trSciInputs = string(sciInputs);
    else
        StrSciInputs = strcat([StrSciInputs, string(sciInputs)], ';');
            // pridani stredniku za kazde cislo
    end
end
mfprintf(u,'\t\t\texprs=['"%s"'],..\n',StrSciInputs)
// zapsani retezce do souboru
```

Výpis 25: Vstup bloku add

Jelikož se nám mění počet vstupních portů, tak musíme do struktury bloku zapisovat hodnoty pro vytvoření a nastavení vstupního portu. Pro vytvoření portu musíme do grafické části bloku zapisovat 'E' jako explicitní port a pro nastavení vstupního portu musíme zapsat do modelové části bloku jako vstup mínus jedničky a jedničky jako typ vstupu. Toto nastavení grafické části bloku můžeme vidět z výpisu 26. Modelová část je velmi podobná jen místo 'E' se v cyklu for nachází 1 nebo -1.

```
for ii = 1:N_Inputs // cyklus od 1 do poctu vstupu
    if ii == 1
        in1_imp = string('E');
    else
        in1_imp = strcat([in1_imp, string('E')], ';');
        // oddeleni E strednikem
    end
end
mfprintf(u,'\t\t\tin_implicit=['"%s'"],..\n',in1_imp)
// zapsani do souboru
```

Výpis 26: Nastavení graficke části bloku pro vstupní port

4.4 Generování bloku GENSIN_f

Blok sine wave se načítá stejně jako blok gain. Opět si zde musíme ošetřit vstup, jelikož když si přidáme v Simulinku blok sine wave a nijak neupravujeme jeho hodnoty, nenačtou se nám do souboru blokdiagram, proto musíme přidat hodnoty amplitude, frequency a phase ručně. Ještě si musíme ošetřit vstup aby se nenačítala hodnota port. Strukturu sine wave vidíme na obrázku 34.

```
BlockType: "Sin"
Name: "Sine Wave"
SID: "13"
Amplitude: "5"
Frequency: "20"
Phase: "10"
Ports: "[0, 1]"
Position: "[110, 160, 140, 190]"
ZOrder: "13"
SampleTime: "0"
obj: 9
```

Obrázek 34: Struktura Sine wave

Opět si pomoci *if exists('Sin') == 1* zjistíme zda existuje struktura Sin, pokud neexistuje tak se do pomocné proměnné uloží číslo objektů gain a add. Pokud existuje uloží se pozice, poté se ze struktury načtou hodnoty pro amplitudu, frekvenci a fázi. Opět se jedná o jednodušší blok, kde se nemění počet portů.

4.5 Generování bloku CMSCOPE

Time Scope neboli osciloskop je nejsložitějším blokem pro převedení. Načítání do struktury je podobné ostatním blokům jen s rozdílem toho, že musíme odfiltrovat atribut *ScopeSpecificationString*. Atribut *ScopeSpecificationString* si do struktury přidáváme pomoci příkazu if, kde si musíme předpřipravit prázdné místo ve struktuře, jinak by se hodnota nepřidala. Právě *ScopeSpecificationString* je textově velice obsáhlý, nese hodnoty všech nastavení grafů a portů. Na strukturu osciloskopu se můžeme vidět na obrázku 35.

```
BlockType: "TimeScope"
Name: "Scope1"
SID: "26"
Ports: "[1]"
Position: "[375, 98, 405, 132]"
ZOrder: "26"
ScopeSpecificationString: "Simulink.scopes.TimeScopeBlockCfg(''CurrentConfiguration'',
NumInputPorts: "1"
obj: 11
MinYlength: "3"
MaxYlength: "6"
```

Obrázek 35: Struktura TimeScope

Opět si pomoci příkazu *if exist* zjistíme zda existuje struktura TimeScope, pokud neexistuje tak se sečtou všechny pomocné z minulých bloků, jelikož tuto hodnotu objektu budeme potřebovat ještě pro spoje. Pokud existuje tak se doplní do souboru pozice, poté se zjistí kolik má osciloskop portů a provede se cyklus for pro vytvoření portů a jejich nastavení. Tento cyklus můžeme vidět z výpisu 27.

```
for qq = 1:N_Ports  //cyklus od 1 do poctu portu
    if qq == 1
        StrPort = string(1); // nastaveni jednicek do portu
        StrPin = string(0); //nastaveni nul pro vstupni porty
        in1sc = string('E');    //nastaveni explicitniho port
        in2sc = string(1);  //nastaveni jednicek vstupniho portu
        refPeriod = string(30); //nastaveni referencni periody
    else
        StrPort = strcat([StrPort, string(1)], ' ');
        //pridani mezer mezi jednicky
        StrPin = strcat([StrPin, string(0)], ';');
        //pridani stredniku mezi nuly
        in1sc = strcat([in1sc, string('E')], ';');
        //pridani stredniku mezi explicitni porty
        in2sc = strcat([in2sc, string(1)], ';');
        //pridani stredniku mezi jednicky
        refPeriod = strcat([refPeriod,string(30)], " ")
```

```
            //pridani mezery mezi periody
        end
    end
```

Výpis 27: Cyklus pro nastavení portů TimeScope

Nyní když jsme si nastavili vstupní porty a referenční periodu, tak jako další potřebujeme vyčíst z již zmiňovaného *ScopeSpecificationString* hodnoty pro Ymin vektor a Ymax vektor. Hodnoty ve *ScopeSpecificationString* jsou uloženy jako typ string, tedy můžeme pro vyhledávaní používat string příkazy *strindex a part*. Jak se vyhledávají potřebné hodnoty pro Ymin vektor můžeme vidět z výpisu 28.

```
    MinY = strindex(TiScSSS(uu), 'MinYLimReal')
    // hledani ve stringu slova MinYLimReal
    MinYlength = length(MinY);  //ulozeni poctu vyskytu slova
    if (MinYlength == N_Ports) then //pokud je pocet slov roven poctu portu
        for ww = 1:MinYlength   //cyklus od 1 do poctu vyskutu slova
            MinYvalue(1,m) = part(TiScSSS(uu), (MinY(ww)+14):((MinY(ww)+16)))
            //zobrazeni souradni, kdy nam vyjde primo hodnota
        end
```

Výpis 28: Cyklus pro vyhledávání Ymin vektor

Jelikož blok pro oscilátor ve Scilab potřebuje mít stejný počet Ymin vektoru jako portů musel se použit proto příkaz *if else*, kde se kontroluje počet portů oproti počtu výskytu slova MinYLimReal. Pokud je počet výskytů slova větší uloží se pouze první hodnoty Ymin vektorů, když je počet výskytů menší doplní se hodnotou -5, jelikož je to základní hodnota. Naprosto stejně je to i pro Ymax vektor jen se pomoci příkazu strindex hledá slovo MaxYLimReal.

4.6 Generování spojů

Nyní si ukážeme jak se načítají spoje do naši struktury viz. obrázek 36. Spoje si načítáme podobně jak bloky, vyhledáváme slovo "Line", poté se nám do struktury uloží hodnoty například *Src:"20#out:1"a Dst:"21#in:2"* což jsou zkratky source a destination. První číslo nám říká SID bloku, dále zda jde o vstupní či výstupní port a poslední číslo je číslo portu.

```
ZOrder: "5"
Src: "20#out:1"
Dst: "21#in:2"
SIDSrc: "20"
SIDDst: "21"
SrcPort: "1"
DstPort: "2"
Points: "[17, 0; 0, -55]"
```

Obrázek 36: Struktura Line

Do naši struktury jsme si tento řetězec rozdělili na jednotlivé hodnoty, které jsou potřeba využít pro tvorbu spoje v Scilab. Toto rozdělení pro source řetězce můžeme vidět z výpisu 29.

```
LineSrc = part(LiSrc, 1:4); // zobrazeni prvnich 4 hodnot
SIDSrcVal = strtod(LineSrc);    // preveni na cisla
SIDSrcValstr = string(SIDSrcVal)    // prevedeni na string
execstr(strcat(['Line(', string(v), ').', 'SIDSrc', ' = ''',SIDSrcValstr ,
    ''''']));
// ulozeni do struktury jako SIDSrc
PortSrcLoc = strindex(LiSrc,':')    // vyhledani dvojtecky
PortSrc = part(LiSrc,PortSrcLoc+1:$)    // zobrazeni cisla
execstr(strcat(['Line(', string(v), ').', 'SrcPort', ' = ''',PortSrc , '''
    ']));
// ulozeni do struktury jako SrcPort$
```

Výpis 29: Rozdělení source řetězce

Stejně jako u bloku si zjistíme pomoci *if exists('Line') == 1* zda spoj existuje, pokud neexistuje soubor se uzavře, program ukončí a otevře se nám okno se zapojením. Pokud existuje zjistí se kolik spojů tam bude, poté se provede cyklus for, který porovnává kolik je v pomocné struktuře hodnot SID. Pomocnou strukturu můžeme vidět na obrázku 37.

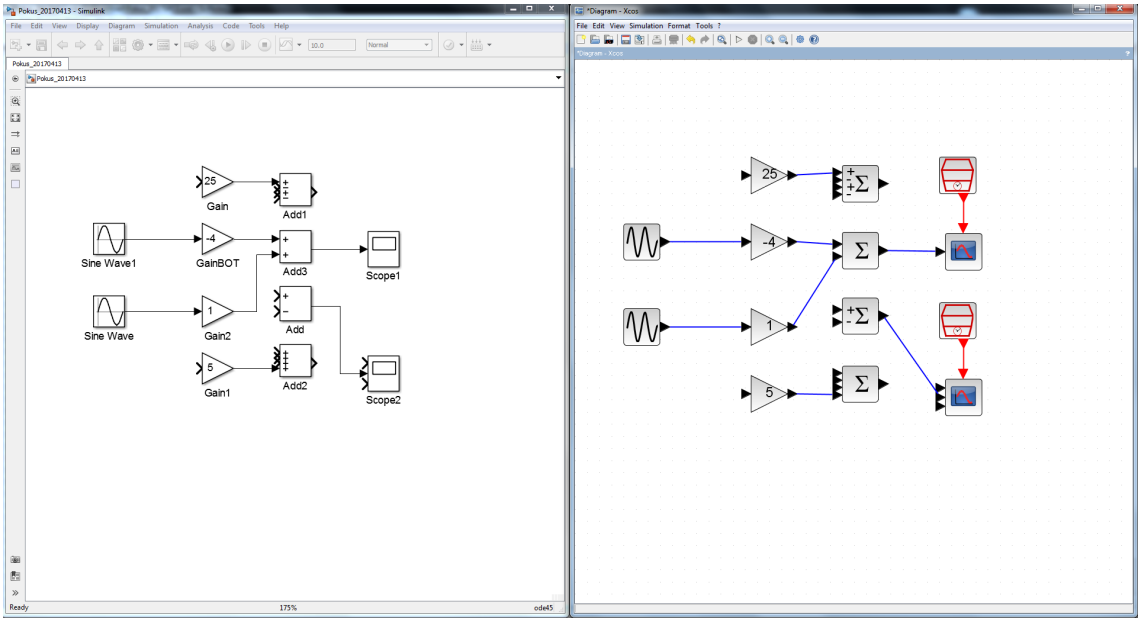
```
12x1 struct array with fields:
   SID
   obj
   Xpos
   Ypos
```

Obrázek 37: Struktura SID

Pomocnou strukturu jsme si vytvořili, aby nám ukládala SID bloku a přiřazovala k nim čísla objektu ve výstupním souboru. Velice lehce si tak můžeme porovnávat SID a číslo objektu, která jsou důležitá pro spoje. Porovnávání SID můžeme vidět z výpisu 30.

```
for cc = 1:length(SID.SID) // cyklus od 1 po pocet SID
    if (SID(cc).SID == Line(gg).SIDSrc)
    //pokud je SID rovne SID linku source
        SrcObj = SID(cc).obj    // vlozeni cisla objektu
    elseif (SID(cc).SID == Line(gg).SIDDst)
    //pokud je SID rovne SID linku dest
        DstObj = SID(cc).obj    // vlozeni cisla objektu
    end
```

Výpis 30: Porovnávání SID a přiřazování čísla objektu



Obrázek 38: Porovnání: vlevo Simulink vpravo Xcos

4.7 Porovnání zkušebního zapojení s převedeným zapojením

Na obrázku 38 můžeme vidět porovnání dvou zapojení, kde na levé straně je zapojení v Simulinku a na pravé straně je vytvořené zapojení v Xcosu. Na první pohled můžeme vidět, že u osciloskopů(Time Scope pro Simulink, CMScope pro Xcos) je v Xcos zapojení navíc SampleCLK blok navíc, jelikož v simulinku osciloskop má clock zabudován v sobě. SampleCLK blok byl přidán ručně, jelikož jeho hodnota ze Simulink souboru není dostupná, může se nacházet v jiných složkách, ale to už je na složitější práce. Dále vidíme ze nejsou udělané body u spojů a uzly. Body a uzly jsou složitější pro převod. Můžeme si povšimnout u sumy (pro Simulink blok add a pro Xcos blok BIGSOM_f), že pokud se znaménka opakují nezobrazují se na bloku.
V této kapitole jsem čerpal z literatury [3] [4] [5]

Závěr

Hlavním cílem této práce bylo vytvoření programu převádějící čtyři zkušební bloky a spoje z licencovaného programu Simulink do volně dostupného programu Xcos. Dalším cílem bylo vytvoření podrobného popisu pracovních ploch programu Scilab a Xcos. Dále pak detailní struktura souboru Xcos a tvorba bloků v jazyce Scilab.

Vytvoření programu proběhlo úspěšně, program funguje pro čtyři zkušební bloky. Vytvoření bloku je náročné, jelikož každý je odlišný a musí se načítat do nové struktury. Program není simulovatelný z důvodu odlišnosti bloku osciloskopu(Scilab potřebuje navíc blok sampleCLK) a nedohledání hodnoty potřebné pro sampleCLK v blockdiagram. Dále pak program nedělá body na spojích, jelikož už je to na větší složitosti dohledaní hodnot.

Podrobný popis pracovních ploch ocení lidé, kteří začínají pracovat s programy pro numerické výpočty a nechtějí platit za drahé licence. V popisu pracovních ploch najdou možnosti pro každé menu, klávesové zkratky a popis tlačítek pro rychlé použití.

Tvorba bloků v jazyce Scilab a jejich hierarchie je potřebná pro pochopení převodu mezi programy a zapisování do souboru.

Literatura

[1] CAMPBELL, S. L., Jean-Philippe. CHANCELIER a Ramine NIKOUKHAH. Modeling and simulation in Scilab/Scicos with ScicosLab 4.4. 2nd ed. New York: Springer, c2010. ISBN 1441955267.

[2] GOMEZ, Claude. Engineering and scientific computing with Scilab. Boston, Mass.: Birkhauser, 1999. ISBN 3764340096.GOMEZ, Claude.

[3] Scicos team. Constructing new blocks in Scicos [online]. 2009. [cit. 2017-04-21]. Dostupné z:http://www.scicos.org/Formation_scicos_mars_2008.pdf

[4] XML Objects. Scilab [online]. [cit. 2017-04-21]. Dostupné z: https://help.scilab.org/docs/6.0.0/en_US/XMLObjects.html

[5] Strings. Scilab [online]. [cit. 2017-04-21]. Dostupné z: https://help.scilab.org/docs/6.0.0/en_US/section_a08ff639ee9ce4bd16aa1c6ac0f86c69.html

Seznam příloh

1block.xcos	Xcos soubor obsahující jeden blok
2block_spoj.xcos	Xcos soubor obsahující dva bloky a spoj
2block_uzel.xcos	Xcos soubor obsahující dva bloky a uzel
SciNotesToXcos.sce	Zdrojový kód ukázkového zapojení s blokem CLOCK_f
SciNotesToXcosSample.sce	Zdrojový kód ukázkového zapojení s blokem sampleCLK
Adresář Prazdne_soubory_xcos	Obsahující 2 složky se 6 prázdnými soubory Xcos, soubory ukazují změnu ID v jednotlivých instancích
Adresář BP	Obsahující bakalářskou práci v latexové podobě
BP_HYK0012.pdf	Bakalářská práce v souboru pdf
Program.sce	Zdrojový kód převádějící bloky ze Simulink do Xcos
Zkusebni_soubor.slx	Zkušební soubor .slx